...e costruisci il tuo LABORATORIO DIGITALE

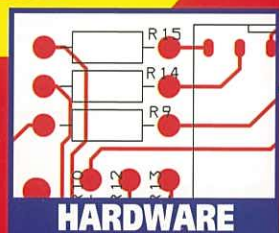
6,90 €

12

HARDWARE

DIGITALE DI BASE

MICROCONTROLLER

DIGITALE AVANZATO

TOTALMENTE PROGRAMMABILE!!!

9 771824 363008 41012

Peruzzo & C.

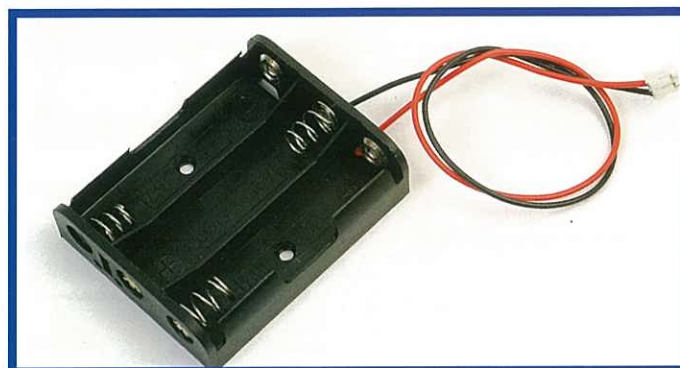
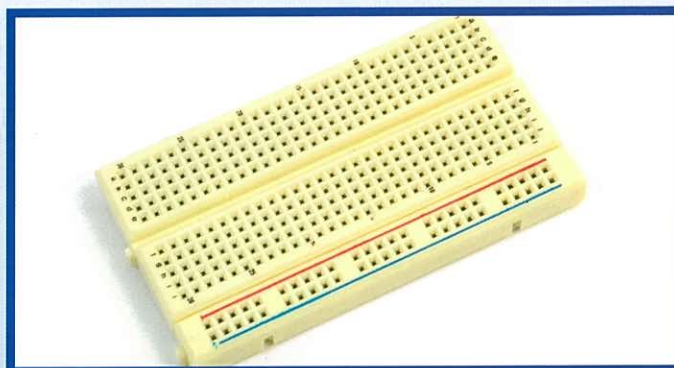
Direttore responsabile:
ALBERTO PERUZZO
Direttore Grandi Opere:
GIORGIO VERCELLINI
Consulenza tecnica e traduzioni:
CONSULCOMP S.n.c.
Pianificazione tecnica
LEONARDO PITTON

Direzione, Redazione, Amministrazione: viale Ercole Marelli 165, Tel. 02/242021, 20099 Sesto San Giovanni (Mi). Pubblicazione settimanale. Registrazione del Tribunale di Monza n. 1738 del 26/05/2004. Spedizione in abbonamento postale gr. II/70; autorizzazione delle Poste di Milano n. 163464 del 13/2/1963. Stampa: Staroffset s.r.l., Cernusco S/N (MI). Distribuzione SO.DI.P. S.p.A., Cinisello Balsamo (MI).

"ELETTRONICA DIGITALE"
si compone di
70 fascicoli settimanali
da suddividere
in 2 raccoglitori.

RICHIESTA DI NUMERI ARRETRATI. Per ulteriori informazioni, telefonare dal lunedì al venerdì ore 9.30-12.30 all'ufficio arretrati tel. 02/242021. Se vi mancano dei fascicoli o dei raccoglitori per completare l'opera, e non li trovate presso il vostro edicolante, potrete riceverli a domicilio rivolgendovi direttamente alla casa editrice. Basterà compilare e spedire un bollettino di conto corrente postale a PERUZZO & C. s.r.l., Ufficio Arretrati, viale Marelli 165, 20099 Sesto San Giovanni (MI). Il nostro numero di c/c postale è 42980201. L'importo da versare sarà pari al prezzo dei fascicoli o dei raccoglitori richiesti, più le spese di spedizione € 3,10 per pacco. Qualora il numero dei fascicoli o dei raccoglitori sia tale da superare il prezzo globale di € 25,82 e non superiore a € 51,65, l'invio avverrà per pacco assicurato e le spese di spedizione ammonteranno a € 6,20. La spesa sarà di € 9,81 da € 51,65 a € 103,29; di € 12,39 da € 103,29 a € 154,94; di € 14,98 da € 154,94 a € 206,58; di € 16,53 da € 206,58 in su. Attenzione: ai fascicoli arretrati, trascorse dodici settimane dalla loro distribuzione in edicola, viene applicato un sovrapprezzo di € 0,52, che andrà pertanto aggiunto all'importo da pagare. Non vengono effettuate spedizioni contrassegno. Gli arretrati di fascicoli e raccoglitori saranno disponibili per un anno dal completamento dell'opera. IMPORTANTE: è assolutamente necessario specificare sul bollettino di c/c postale, nello spazio riservato alla causale del versamento, il titolo dell'opera nonché il numero dei fascicoli e dei raccoglitori che volete ricevere.

IN REGALO in questo fascicolo

1 Portabatterie con connettore femmina a 2 poli

IN REGALO nel prossimo fascicolo

Modulo "Bread Board" per prove

COME RACCOGLIERE E SUDDIVIDERE L'OPERA NELLE 4 SEZIONI

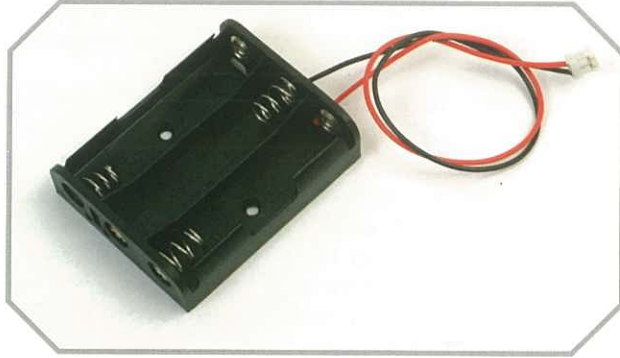
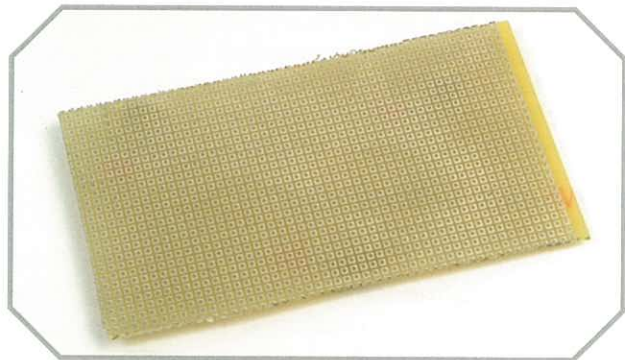
L'Opera è composta da 4 sezioni identificabili dalle fasce colorate, come indicato sotto. Le schede di ciascun fascicolo andranno suddivise nelle sezioni indicate e raccolte nell'apposito raccoglitore, che troverai presto in edicola. Per il momento, ti consigliamo di suddividere le sezioni in altrettante cartellette, in attesa di poterle collocare nel raccoglitore. A prima vista, alcuni numeri di pagina ti potranno sembrare ripetuti o sbagliati. Non è così: ciascuno fa parte di sezioni differenti e rispecchia l'ordine secondo cui raccogliere le schede. **Per eventuali domande di tipo tecnico scrivere al seguente indirizzo e-mail:** elettronicadigitale@microrobots.it

Hardware Montaggio e prove del laboratorio

Digitale di base Esercizi con i circuiti digitali

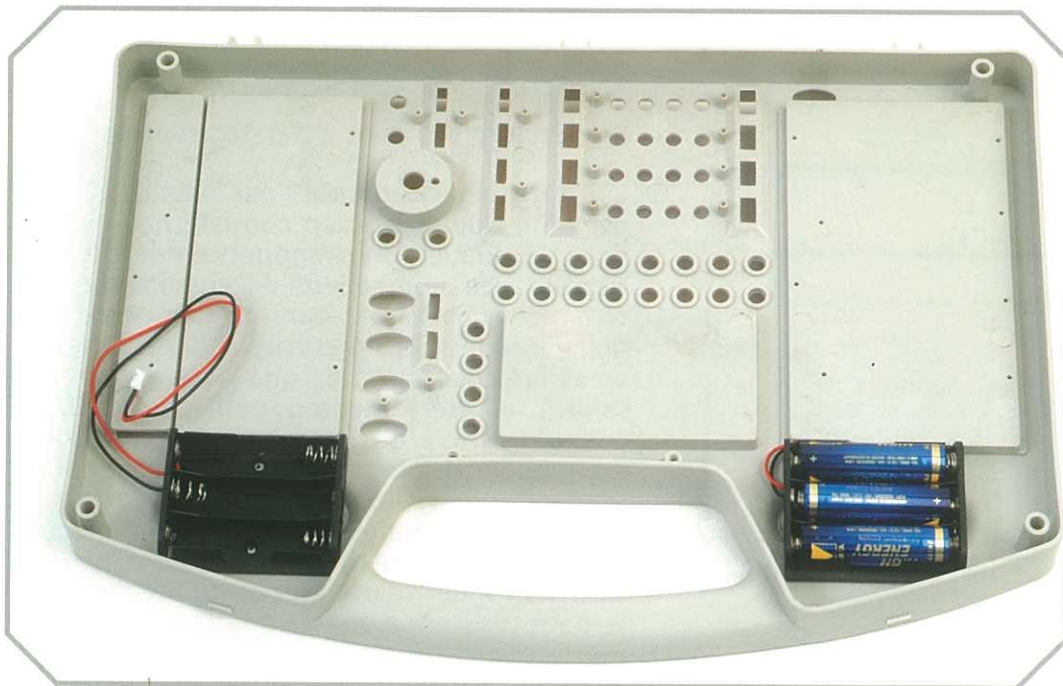
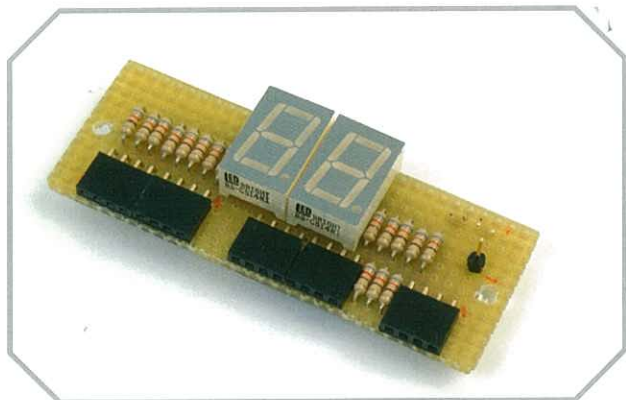
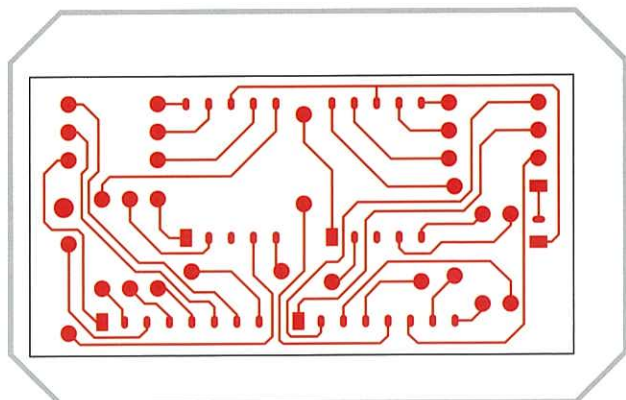
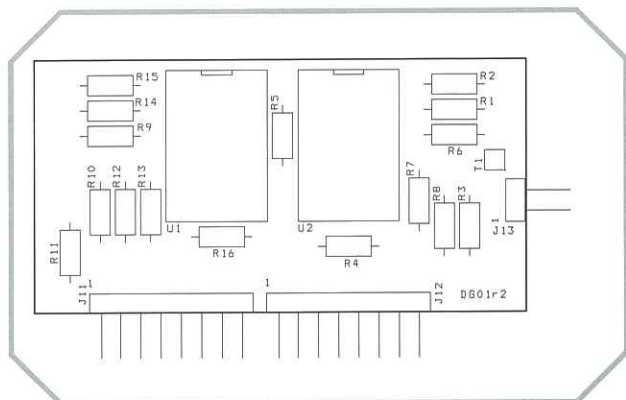
Digitale avanzato Esercizi con i circuiti sequenziali

Microcontroller Esercizi con i microcontroller

Alimentazione portabatterie

Portabatterie con il suo connettore.

Scheda mille fori per la costruzione di prototipi.

In questo numero vi viene fornito il secondo portabatterie che è uguale al precedente e può contenere tre pile del tipo AA, chiamate anche R6, che, collegate in serie, permettono di ottenere una tensione nominale di 4,5 V. Non lo inseriremo nella sua posizione definitiva perché lo utilizzeremo per alimentare alcuni circuiti fino a quando non avremo il circuito stampato di distribuzione dell'alimentazione. Riguardo al portabatterie non c'è molto da dire, quindi spiegheremo in modo rapido e semplice i modelli più comuni di circuiti stampati.

Collegamento di componenti

Un circuito elettronico è formato da diversi componenti collegati fra loro in modo affidabile e sicuro, sia dal punto di vista elettrico che meccanico. Esistono diversi modi di collegare componenti, parleremo di quelli che fissano i componenti in modo definitivo, lasciando a parte i circuiti sperimentali a inserzione meccanica in cui è possibile recuperare la totalità dei

Vista del laboratorio in cui è possibile notare la posizione dove verranno installati due portabatterie.

Prototipo realizzato con una scheda mille fori.

Disegno di piste di un circuito stampato.

Serigrafia di un circuito stampato.

componenti. Il metodo più conosciuto e pratico consiste nell'utilizzare un circuito stampato, di cui esistono diversi modelli.

Dobbiamo fare una serie di considerazioni in base all'uso e all'ubicazione del circuito stampato. In un'officina la temperatura può variare tra 10° e 40° C nei casi più estremi, inoltre, a parte piccoli colpi casuali ricevuti durante il trasporto, il dispositivo resta immobile. Tuttavia se pensiamo, ad esempio, al circuito stampato dell'accensione elettronica di un'automobile, questo deve sopportare temperature estreme che possono oscillare facilmente tra i -10° e +75° C, inoltre è sottoposto a forti movimenti e vibrazioni. In questi casi estremi occorre fare una serie di considerazioni termiche, dato che il circuito si può deteriorare per eccesso di temperatura di alcuni componenti o per problemi di dilatazione. Per quanto riguarda i colpi e le vibrazioni, potrebbe anche essere necessario fissare singolarmente alcuni componenti.

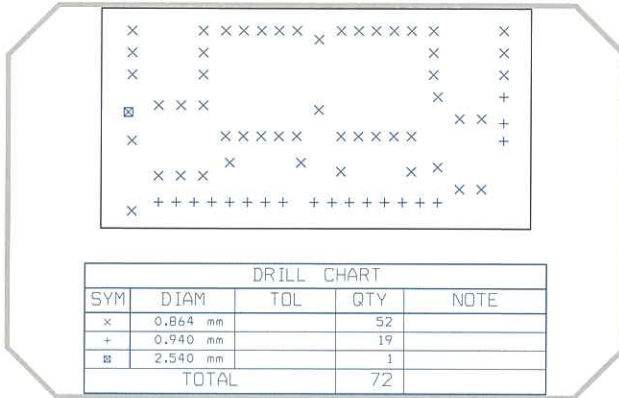
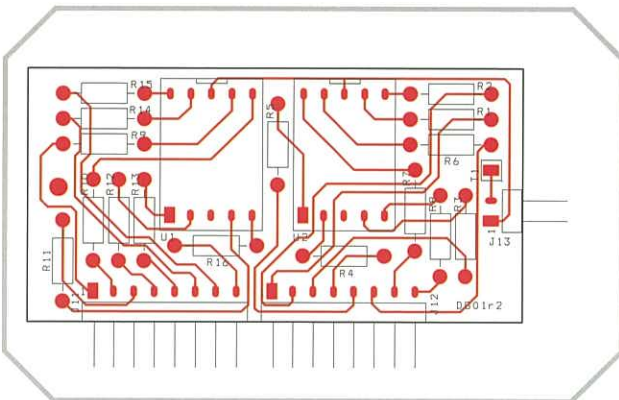
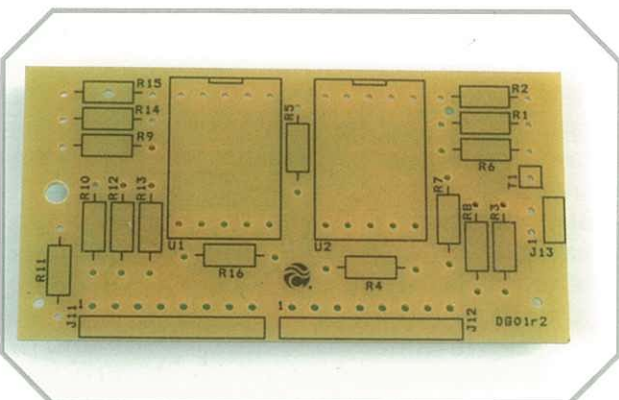
Circuito stampato a faccia singola

Fondamentalmente un circuito stampato consiste in un supporto isolante, normalmente di uno spessore di circa 1,5 mm, formato da fibra di vetro agglomerata con resina epossidica e delle piste di rame che aderiscono a una delle sue superfici, le quali si utilizzano come dei conduttori per collegare tra loro i diversi componenti del circuito; i collegamenti tra i terminali e le piste di rame si realizzano tramite saldatura con lega di stagno.

Montaggio dei componenti

Ci sono due procedimenti per installare i componenti, quello classico consiste nell'inserire ogni terminale del componente nel foro corrispondente della scheda del circuito stampato e realizzare la saldatura dal lato posteriore della scheda sul foro corrispondente, che deve essere circondato da una pista di rame. La saldatura fisserà il componente e stabilirà il collegamento elettrico, è possibile realizzarla manualmente o in modo automatico mediante macchine di saldatura a onda di stagno.

Il circuito stampato è un sistema per montaggio definitivo di componenti, non è progettato per togliere e mettere componenti, anche se deve resistere a questa operazione per permettere la sostituzione dei componenti durante la fase di messa a punto o di riparazione.

DRILL CHART				
SYM	DIAM	TOL	QTY	NOTE
×	0.864 mm		52	
+	0.940 mm		19	
⊠	2.540 mm		1	
TOTAL			72	

Piano di foratura.

Disegno completo del circuito stampato.

Risultato reale del disegno precedente.

SMD

L'altra tecnica consiste nel montaggio superficiale dei componenti; essi vengono sistemati sulle piste in modo che i terminali coincidano con le zone di rame prestagnato predisposte per la saldatura in cui avverrà direttamente il collegamento. Si tratta di una tecnica molto complessa, in cui la saldatura manuale è utilizzata solamente per alcune riparazioni o per i prototipi. La saldatura viene fatta normalmente in modo automatico e vi sono tre tecniche principali: onda, infrarossi o stadio a vapore. Permette di utilizzare componenti con un'altissima densità di terminali, tanto che a prima vista può sembrare quasi impossibile.

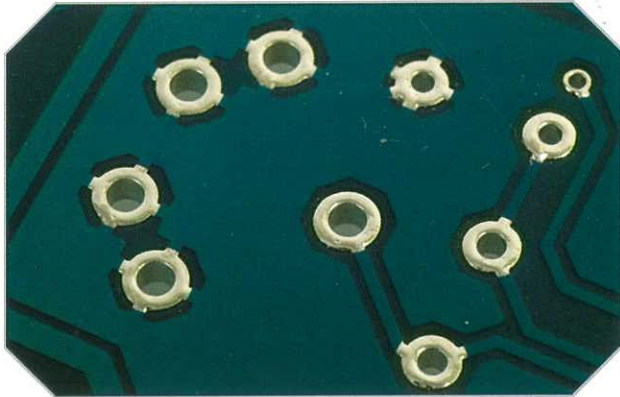
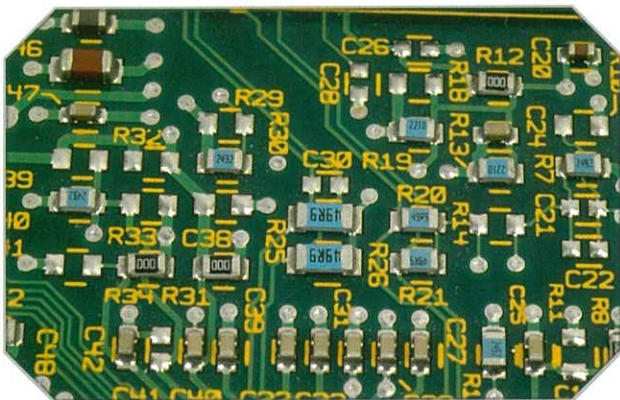
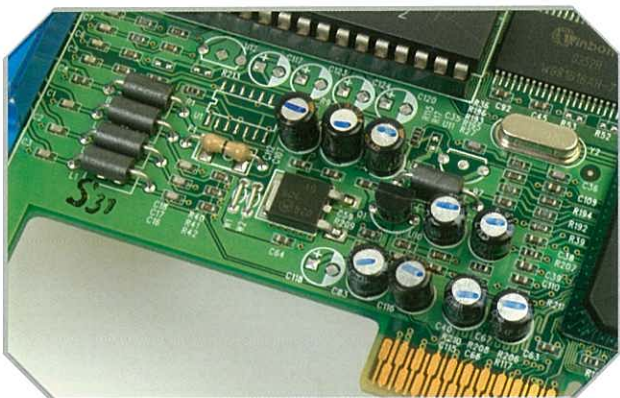
Costruzione del circuito stampato

Il circuito stampato viene costruito sulla base di un disegno preesistente e si utilizzano tecniche fotografiche e di attacco chimico. Si parte da un supporto isolante con una lamina di rame incollata su uno dei lati, per quanto riguarda i circuiti a singola faccia, e con una lamina per ogni lato nel caso di quelli a doppia faccia.

Normalmente il primo passo è la realizzazione di fori sulla scheda. In seguito, tramite erosione chimica, si elimina la parte necessaria alla formazione delle piste. La protezione delle zone che non devono essere attaccate viene fatta con resine fotosensibili. A questo punto si realizza la metallizzazione dei fori in modo da mettere in collegamento entrambi i lati; in questo processo si ricopre tutta la superficie di rame con stagno per facilitare le operazioni di saldatura.

Il passo successivo è l'applicazione, per mezzo di tecniche di stampa, della serigrafia nello strato di "solder resist", che fondamentalmente è una vernice isolante e resistente alla saldatura che ricopre tutta la superficie della scheda, a eccezione delle zone di rame dove vogliamo fare attaccare lo stagno. Il lato componenti deve ospitare lo strato di "solder resist", oltre alla serigrafia che ci indica dove posizionare i componenti, facilitandone la localizzazione durante i lavori di riparazione e di regolazione.

Circuito stampato universale

Per la realizzazione di prototipi di circuiti semplici esiste una serie di modelli di circuiti

PCB a doppia faccia con fori metallizzati.

PCB con componenti SMD.

PCB misto, con componenti SMD e convenzionali.

stampati con molti fori. Normalmente in questi circuiti ogni foro occupa l'incrocio delle linee di un reticolo di un decimo di pollice per un decimo di pollice, in modo da permettere l'inserzione diretta dei circuiti integrati. Il cablaggio si può realizzare con filo isolato saldabile; vi sono alcuni modelli per questo tipo di lavori in cui non è necessario togliere la parte isolante per eseguire la saldatura.

Questo procedimento è molto veloce dato che ci permette di disporre del circuito immediatamente, però normalmente si utilizza solo per circuiti con un basso numero di componenti.

Circuito stampato a doppia faccia

Questo tipo di circuito è il più utilizzato. Ha piste di rame su entrambi i lati e la giunzione tra essi è realizzata tramite dei fori metallizzati. Quando si utilizzano componenti SMD è possibile collocarli su entrambi i lati. Se il montaggio è misto, i componenti classici con terminali per i fori vengono montati su un solo lato.

Circuiti multi-strato

Questi circuiti hanno diversi strati di piste di rame, cosa che facilità un'alta densità di collegamenti. Normalmente si utilizzano nelle schede madri dei computer e in sistemi altamente professionali dato che il loro impiego permette di utilizzare frequenze molto elevate nei circuiti stampati.

Montaggio misto

Il montaggio misto è largamente utilizzato, perché alcuni componenti non sono facili da reperire nella versione per montaggio superficiale o per altre ragioni, principalmente economiche, che ne consigliano l'uso. Gli elementi voluminosi e di collegamento sono montati a mano.

Bisogna tener presente che il montaggio superficiale è adatto all'automatizzazione, ottenendo circuiti di elevata qualità. A partire da medi volumi di produzione è molto redditizio, la dimensione del dispositivo si riduce in modo considerevole, inoltre risultano abbastanza robusti da un punto di vista meccanico.

Simboli e schemi

Quando i circuiti elettronici erano semplici ed erano costruiti con elementi discreti, si rappresentavano tutti i componenti. In seguito arrivarono i primi circuiti integrati che contenevano diversi transistor al loro interno e che consistevano, in realtà, in un amplificatore operazionale, cambiando la rappresentazione degli schemi con blocchi funzionali, ovvero circuiti che realizzano una funzione specifica e hanno ingressi e uscite.
A volte, per eliminare linee di disegno dagli schemi, non vengono rappresentati i terminali e i collegamenti dell'alimentazione.

Circuiti digitali

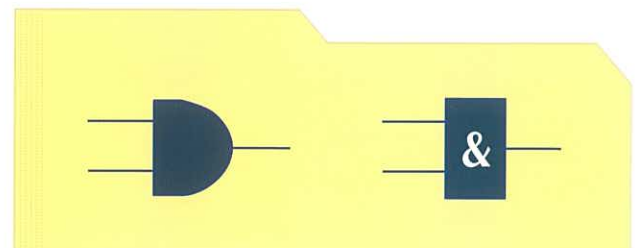
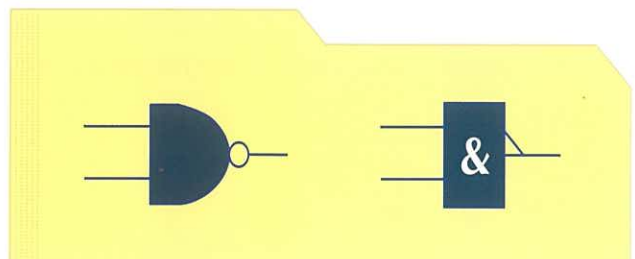
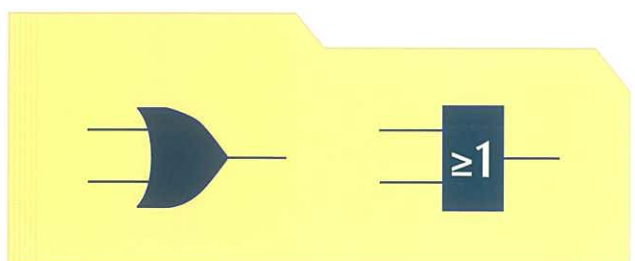
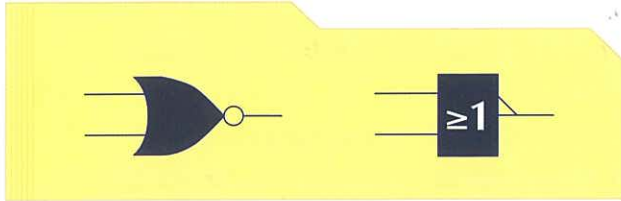
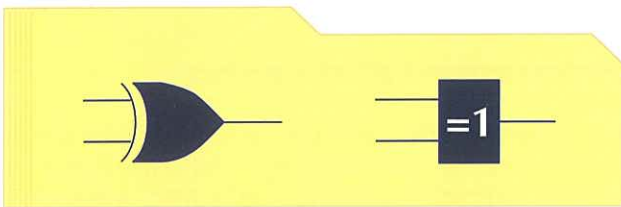
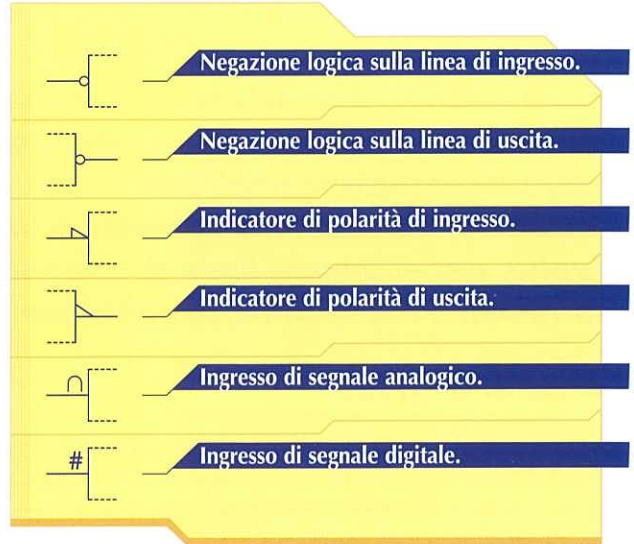
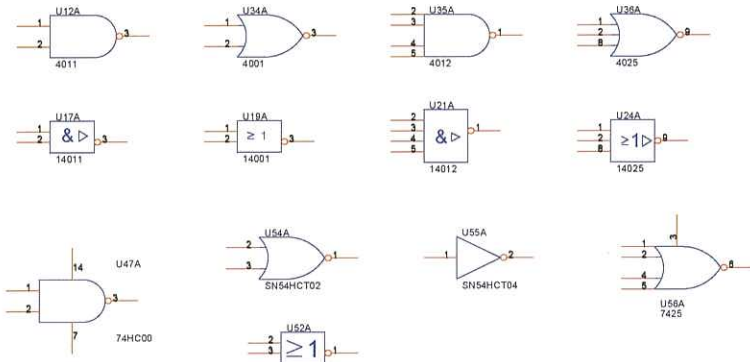
Nei circuiti digitali si utilizzano normalmente i diagrammi logici e la rappresentazione delle porte che già conosciamo come classica, tuttavia sempre più spesso sono utilizzati i simboli di rappresentazione delle norme IEC (International Electrotechnical Commision), che sono in realtà molto valide ma, se siamo abituati alla rappresentazione classica, ci possono sembrare all'inizio un po' più complicate. Pertanto dobbiamo conoscere alcuni aspetti fondamentali per poter interpretare correttamente gli schemi.

Simbolo

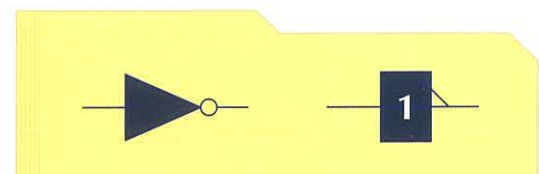
I simboli più semplici sono composti da un corpo rettangolare, gli ingressi a sinistra e le uscite a destra. Su ogni linea di ingresso e di uscita sono indicate le polarità. All'interno del corpo viene posizionato il simbolo che indica la funzione fondamentale.

SIMBOLI GENERALI	
=	Funzione logica identità
&	Funzione logica AND
>1	Funzione logica OR
=1	Funzione logica OR ESCLUSIVO
COMP	Comparatore
Σ	Sommatore (ADDER)
P-Q	Sottrattore
II	Moltiplicatore
MUX	Multiplexer
DX	Demultiplexer

Programmi di disegno

I programmi utilizzati per la costruzione degli schemi hanno normalmente delle librerie con diversi sistemi di simboli e, di solito, è possibile lavorare con qualsiasi di essi indifferentemente.

Simboli logici e IEC della porta AND.

Simboli logici e IEC della porta NAND.

Simboli logici e IEC della porta OR.

Simboli logici e IEC della porta NOR.

Simboli logici e IEC della porta OR ESCLUSIVO.

Ingressi e uscite più comuni.

Simboli utilizzati in programmi di disegno per porte logiche.

Blocchi di controllo comuni che indicano un ingresso comune a diversi blocchi.

Esempi di simboli logici e IEC.

Simboli logici e IEC della porta invertente.

Progetto con MPLAB

Per conoscere meglio MPLAB faremo un programma e penseremo a cosa fare con esso. Questo potente strumento permette la scrittura, la compilazione e la simulazione di un progetto. Con MPLAB voi potrete analizzare come interagisce il nostro programma con il microcontroller, come si gestisce la memoria, quali risorse utilizza, ecc. I lavori di correzione degli errori e di messa a punto risulteranno più semplici utilizzando questo strumento. In questo capitolo ci concentreremo sulla creazione del programma, lasciando la compilazione e la simulazione ai capitoli successivi.

Definizione del progetto

Abbiamo già visto come creare un progetto in MPLAB, ma faremo comunque un rapido ripasso per poter essere sicuri di eseguire tutti i passaggi necessari. Nel menù di controllo selezioneremo l'opzione File → New, o attiveremo l'icona Crea nuovo documento sulla barra degli strumenti. Daremo poi un nome al nostro progetto e cliccheremo due volte OK, dopodiché apparirà la finestra dell'editor.

Siamo pronti per scrivere il nostro primo programma. Il linguaggio che utilizzeremo sarà l'assembler. Il nostro programma farà la somma di due valori diretti, il valore esadecimale 0x05 con 0x07, e il risultato della somma verrà caricato all'indirizzo della memoria dei dati 0x20. Tutti i file che contengono un codice in assembler avranno l'estensione ".asm".

Lavoriamo con MPLAB.

Concetti generali di programmazione

Sappiamo quanto sia importante una corretta metodologia di programmazione, per questo prima di iniziare a scrivere il nostro codice studieremo le regole fondamentali di programmazione in assembler.

La prima colonna dell'editor è riservata alle etichette. Le etichette sono espressioni alfanumeriche scelte dal programmatore per definire valori di indirizzi di memoria. Dovranno iniziare sempre con una lettera e non potranno essere usate espressioni riservate dell'assembler, come:

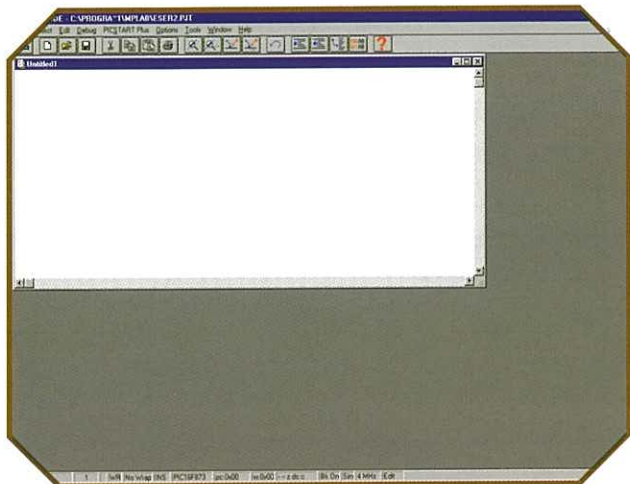
- Istruzioni.
- Direttive.
- Nomi dei registri speciali o dei bit che ne fanno parte.

Nelle colonne successive potremo scrivere il codice mnemonico delle istruzioni o le direttive dell'assembler.

È assolutamente necessario includere commenti al programma per poterlo capire. Quando MPLAB trova un punto e virgola ";" su una linea, considera tutto ciò che viene dopo un commento. Non genererà codice macchina e non saranno considerati gli spazi bianchi o i simboli utilizzati.

È consigliabile accedere a ogni campo utilizzando il tabulatore (separeremo le colonne mediante tabulazioni), in questo modo otterremo un programma più facile da capire.

Per utilizzare maiuscole e minuscole si seguono delle regole ben precise:

- Le direttive dell'assembler si scrivono in maiuscolo.
- I nomi delle variabili si scrivono in maiuscolo.

Editor di testo dove scrivere il nostro codice.

- Gli mnemonici delle istruzioni si scrivono in minuscolo.

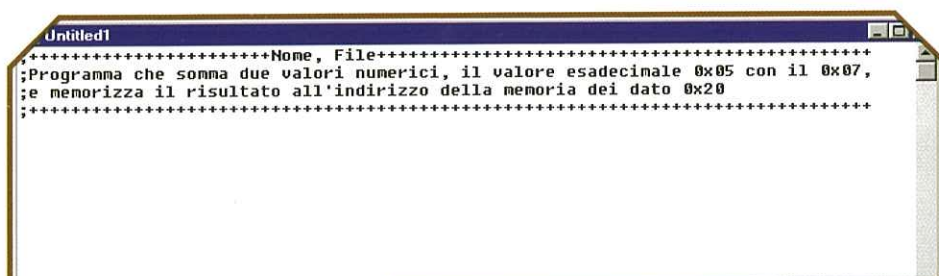
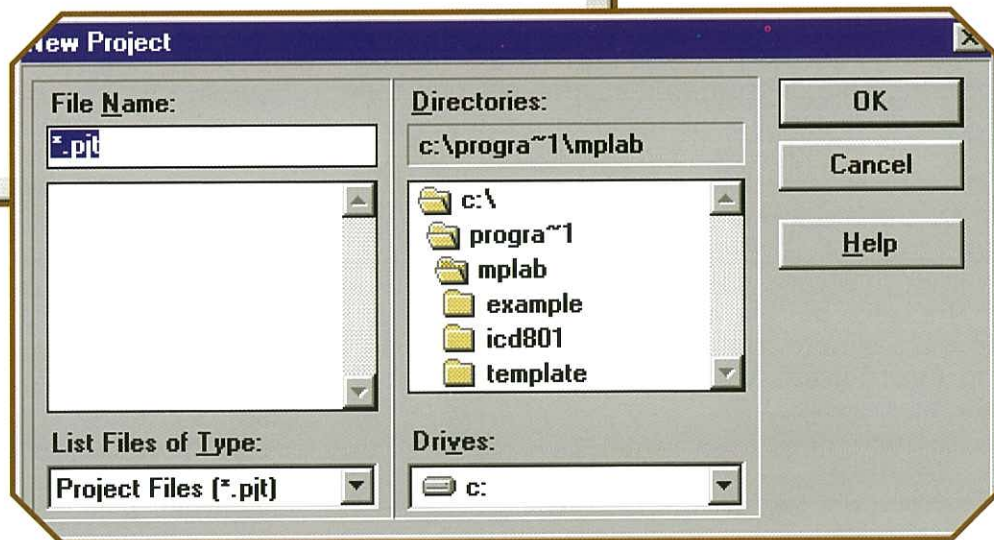
Queste regole non sono obbligatorie. Ogni programmatore ha le sue manie o le sue abitudini, però dovrà seguire una metodologia per fare in modo che i suoi programmi siano leggibili da qualsiasi altro programmatore.

Scrittura del codice

Il nostro programma inizierà con la spiegazione di ciò che si vuole fare, quindi dovremo utilizzare dei commenti. Normalmente il programmatore pone all'interno del programma il suo nome e la data in cui è stato realizzato. Tra i commenti potremo anche trovare il microcontroller utilizzato, la sua configurazione, ecc.

Mediante la direttiva LIST definiremo il microcontroller PIC che utilizzeremo nel progetto e con l'istruzione "include" includeremo nel programma un file che è una libreria dove vengono definite tutte le etichette di tutti i registri del PIC.

Queste saranno sempre le nostre due prime linee di codice in qualsiasi programma. Dovremo sempre dire al compilatore il PIC con cui vogliamo lavorare e inserire una libreria dove siano definiti i nomi di tutti i registri e i loro bit. In questo modo potremo far riferimento a essi utilizzando il nome invece dell'indirizzo che occupano nella memoria. Questo file si dovrà trovare nella directory in cui salveremo il nostro progetto, dato che il compilatore cerca in

```
;++++++++++++++++++++++++Nome, File++++++++++++++++++++++++++++++++++++++++++++++++++
;Programma che somma due valori numerici, il valore esadecimale 0x05 con il 0x07,
;e memorizza il risultato all'indirizzo della memoria dei dato 0x20
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Primi passi: indichiamo ciò che deve fare il nostro programma.

Creazione di un nuovo progetto.

```
:\progra~1\mplab\ed12_1.asm
;+++++++++++++++++++++++Nome, File++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;Programma che somma due valori numerici, il valore esadecimale 0x05 con il 0x07,e memorizza
;il risultato all'indirizzo della memoria dei dato 0x20
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                LIST    p=16F870          ;Definizione del microcontroller
                include "P16F870.inc"     ;Libreria con le etichette di tutti i registri

;Dichiarazione di variabili

RISULTATO       EQU     0x20              ;L'etichetta risultato si associa all'indirizzo 0x20
```

Stato attuale del nostro programma.

```
:\progra~1\mplab\ed12_1.asm
;+++++++++++++++++++++++Nome, File++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;Programma che somma due valori numerici, il valore esadecimale 0x05 con il 0x07,e memorizza
;il risultato all'indirizzo della memoria dei dato 0x20
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                LIST    p=16F870          ;Definizione del microcontroller
                include "P16F870.inc"     ;Libreria con le etichette di tutti i registri

;Dichiarazione di variabili

RISULTATO       EQU     0x20              ;L'etichetta risultato si associa all'indirizzo 0x20

                ORG     0x00              ;Vector di reset. L'istruzione successiva si carica
                                          ;all'indirizzo 0x00
                goto    INIZIO            ;Salto all'istruzione etichettata come INIZIO

                ORG     0x05              ;Istruzione successiva su 0x05, Vector di Interrupt

INIZIO:         movlw   0x05              ;Muove sul registro W il valore numerico 0x05
                addlw   0x07              ;Somma al contenuto di W il valore 0x07, depositando
                                          ;il risultato su W
                movwF   RISULTATO         ;Sposta il contenuto di W sulla variabile RISULTATO
```

Il programma praticamente terminato.

questa directory tutti i file che sono stati inclusi nel programma da compilare.

Potete aprire il file "P16F870.inc" che si trova sul CD e osservare come vengono definiti i registri.

Dopo queste due linee di codice abbiamo la zona delle dichiarazioni delle variabili. Prima di iniziare il programma, dobbiamo dichiarare tutte le variabili che utilizzeremo all'interno dello stesso. Per questo dobbiamo nominarle e assegnare loro una zona di memoria. Nel nostro progetto dobbiamo definire la variabile risultato, dove scriveremo il risultato dell'operazione. Poiché l'indirizzo della memoria ci è stato imposto dalle specifiche di progetto, non ci resta che mettere in relazione il nome con l'indirizzo e questo si fa con la direttiva EQU.

Nella figura possiamo osservare la forma che sta acquisendo il nostro programma.

Ciò che viene di seguito verrà utilizzato anche in tutti i programmi che faremo. Bisogna definire dove si troverà il codice da eseguire e in quale zona della memoria di codice si trova la routine di interrupt, nel caso sia presente.

Quindi, mediante la direttiva ORG, definiamo la posizione iniziale del codice del nostro programma. Con l'istruzione di salto goto indichiamo dove deve essere la destinazione del salto incondizionato.

A partire dall'etichetta di INIZIO potremo scrivere il codice riferito alle specifiche

```
        IFNDEF __16F870
        MESSG "Processor-header file mismatch.  Verify selected processor."
        ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                            EQU     H'0000'
F                            EQU     H'0001'

;----- Register Files------------------------------------------------------

INDF                         EQU     H'0000'
TMR0                         EQU     H'0001'
PCL                          EQU     H'0002'
STATUS                       EQU     H'0003'
FSR                          EQU     H'0004'
PORTA                        EQU     H'0005'
PORTB                        EQU     H'0006'
PORTC                        EQU     H'0007'
PCLATH                       EQU     H'000A'
INTCON                       EQU     H'000B'
PIR1                         EQU     H'000C'
PIR2                         EQU     H'000D'
TMR1L                        EQU     H'000E'
TMR1H                        EQU     H'000F'
T1CON                        EQU     H'0010'
TMR2                         EQU     H'0011'
T2CON                        EQU     H'0012'
CCPR1L                       EQU     H'0015'
CCPR1H                       EQU     H'0016'
CCP1CON                      EQU     H'0017'
RCSTA                        EQU     H'0018'
TXREG                        EQU     H'0019'
RCREG                        EQU     H'001A'
ADRESH                       EQU     H'001E'
ADCON0                       EQU     H'001F'

PTION_REG                    EQU     H'0081'
SA                           EQU     H'0085'
                             EQU     H'0086'
                             EQU     H'0087'
```

Libreria "P16F870.inc" dove si trovano le definizioni di tutti i registri.

Salviamo il codice.

```
...ve File As
File Name:            Directories:                       OK
*.c;*.asm             c:\progra~1\mplab\progetti         Cancel
eser2.asm             c:\                                Help
prova1.asm              progra~1                         [ ] UNIX format
prova2.asm                mplab                          [x] Keep backup
prova3.asm                  progetti
List Files of Type:   Drives:
Source files (*.c;*.asm)   c:
```

```
...progra~1\mplab\ed12_1.asm
;+++++++++++++++++++++Nome, File+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;Programma che somma due valori numerici, il valore esadecimale 0x05 con il 0x07,e memorizza
;il risultato all'indirizzo della memoria dei dato 0x20
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            LIST    p=16F870          ;Definizione del microcontroller
            include "P16F870.inc"     ;Libreria con le etichette di tutti i registri

;Dichiarazione di variabili

RISULTATO   EQU     0x20              ;L'etichetta risultato si associa all'indirizzo 0x20

            ORG     0x00              ;Vector di reset. L'istruzione successiva si carica
                                      ;all'indirizzo 0x00
            goto    INIZIO            ;Salto all'istruzione etichettata come INIZIO

            ORG     0x05              ;Istruzione successiva su 0x05, Vector di Interrupt

INIZIO:     movlw   0x05              ;Muove sul registro W il valore numerico 0x05
            addlw   0x07              ;Somma al contenuto di W il valore 0x07, depositando
                                      ;il risultato su W
            movwf   RISULTATO         ;Sposta il contenuto di W sulla variabile RISULTATO

STOP        nop                       ;Non fa niente però riserviamo queste posizioni per
            nop                       ;collocare un punto di arresto

            END                       ;Direttiva che indica la fine del programma sorgente
```

Codice finale risultante.

del progetto. Anche se non conosciamo il repertorio delle istruzioni del nostro microcontroller, possiamo risolvere il programma con tre sole istruzioni. Mediante l'istruzione movlw carichiamo un valore sul registro di lavoro W, con l'istruzione addlw sommiamo il valore indicato nell'operando al registro di lavoro. Il risultato dell'operazione viene scritto nel registro W, sovrascrivendo il valore precedente. Dobbiamo quindi spostare il risultato alla variabile che abbiamo definito e, a questo scopo, utilizzeremo l'istruzione movwf.

Nella figura possiamo vedere come diventa il programma. Non ci resta che terminarlo, ovvero trasmettere al microcontroller l'ordine di terminare l'esecuzione del codice. I programmi si terminano con la direttiva END. Per una successiva analisi del programma inseriremo delle istruzioni che non fanno nulla, nop (No OPeration), in questo modo potremo simulare il programma e verificare che funzioni realmente come desideriamo. Potremmo anche utilizzare un'istruzione di sleep in modo che il microcontroller, o il simulatore nel nostro caso, si fermi in stato di riposo o addormentato.

Il programma risultante si può vedere nella figura.

Che cosa facciamo con il programma?

Dopo aver terminato il programma, per prima cosa lo dobbiamo salvare. Conosciamo già questo processo, ma vale la pena di ripassarlo. Selezioneremo File e Save e apparirà la finestra di dialogo della figura, dove daremo un nome al file creato. Possiamo chiudere il programma, salvando i cambiamenti fatti nel progetto precedente, dato che per il momento non lavoreremo più con esso.

Nei capitoli successivi faremo pratica con questo programma con le opzioni di compilazione e simulazione. Compilando si convertirà il codice creato in assembler in codice macchina e, nel frattempo, scopriremo l'eventule presenza di errori nel nostro codice. Simulando potremo verificare che il programma creato risponda alle esigenze iniziali del progetto. Nei fascicoli successivi lavoreremo con queste opzioni e potremo verificare quanto sia semplice gestire questo software e quanto risulti utile.

L'obiettivo è imparare a utilizzare questo potente strumento, dato che risulta fondamentale la programmazione dei microcontroller PIC.

La porta C

La porta C completa l'analisi di uno dei dispositivi più importanti del PIC16F870, le porte di I/O. Inoltre avanziamo nel vasto campo della programmazione analizzando i possibili errori che potremmo commettere al momento di confezionare un programma.

La porta C* dispone di otto linee bidirezionali. Per indicare se i terminali lavorano come ingressi o come uscite si utilizza il registro TRISC. Attivando un bit del registro TRISC si indica quindi che il terminale corrispondente della porta verrà utilizzato come ingresso, e se imposteremo a 0 un bit del registro TRISC il terminale corrispondente del registro PORTC rimarrà configurato come uscita.

Il PIC16F870 dispone di tre porte, A, B e C (altri modelli superiori possono avere anche le porte D ed E). Studiando le due porte precedenti abbiamo potuto verificare come le differenze tra le porte siano radicate nelle funzioni che hanno multiplexate e nell'architettura interna delle stesse. Con la porta C avviene la stessa cosa, il funzionamento di base è uguale, cambia però la possibilità di multiplexare le sue funzioni di I/O di dati digitali con altre, necessarie per altri dispositivi del microcontroller.

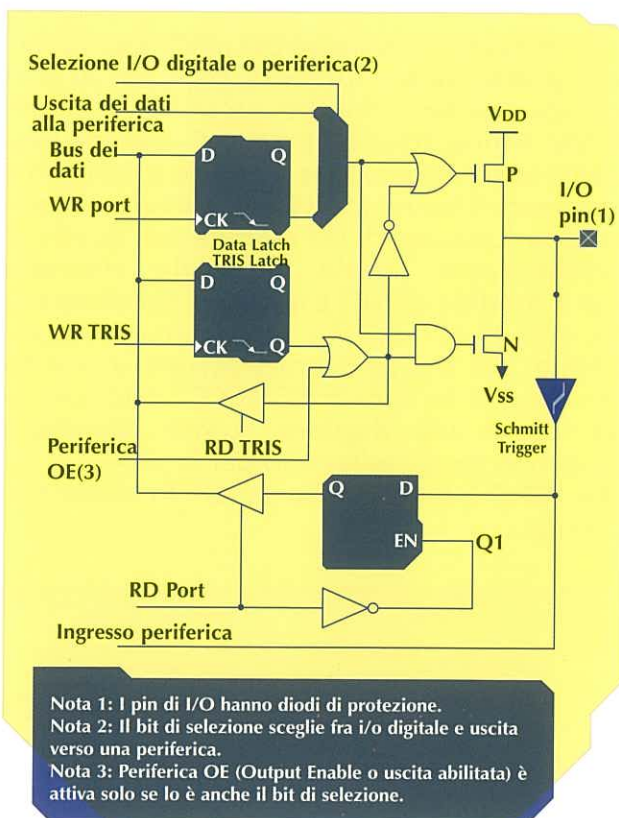
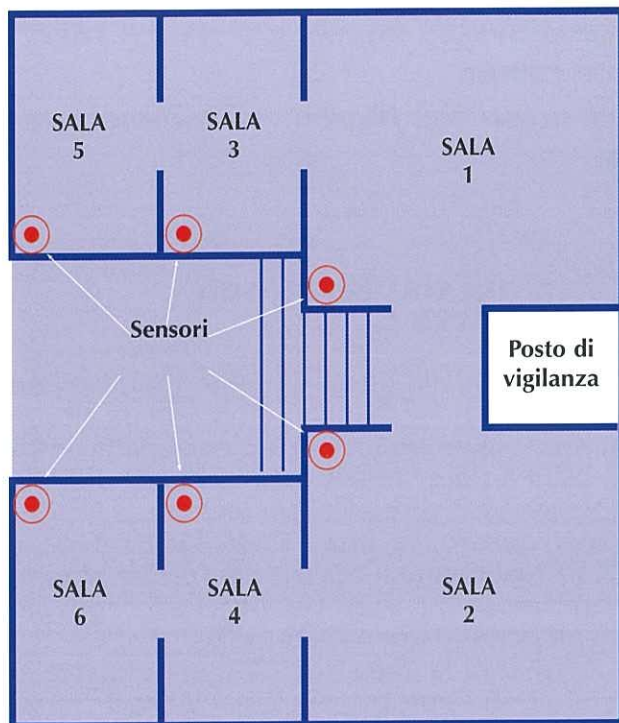
Nome	Bit #	Buffer	Funzione
RC0/T1OSO/T1CKI	bit 0	ST	Ingresso/Uscita digitale o uscita dell'oscillatore del TMR1 o ingresso del clock per il TMR1
RC1/T1OSI	bit 1	ST	Ingresso/Uscita digitale o ingresso dell'oscillatore del TMR1
RC2/CCP1	bit 2	ST	Ingresso/Uscita digitale o ingresso capture/uscita compare/uscita PWM
RC3	bit 3	ST	Ingresso/Uscita digitale
RC4	bit 4	ST	Ingresso/Uscita digitale
RC5	bit 5	ST	Ingresso/Uscita digitale
RC6/TX/CK	bit 6	ST	Ingresso/Uscita digitale o Trasmissione USART asincrona o clock modo sincrono
RC7/RX/DT	bit 7	ST	Ingresso/Uscita digitale o Ricezione USART asincrona o dati modo sincrono

Nome e funzione dei terminali della porta C.

Funzioni dei terminali della porta C

I terminali della porta C hanno multiplexate diverse funzioni che sono utilizzate da alcune periferiche interne. Cinque piedini della porta C, oltre a poter funzionare come I/O digitali, possono assolvere ad altre funzioni.

Per questo, le linee RC0/T1OSO/T1CKI e RC1/T10SI hanno il compito di fornire al temporizzatore 1 (Timer 1 o TMR1) dei collegamenti richiesti verso l'esterno.

Mediante la linea RC2/CCP1 si permette al modulo di Capture/Compare/PWM di cui dispone il PIC16F870, di poter interagire con l'esterno. Questa linea permetterà al disposi-

Architettura interna dei terminali della porta C.

Un progetto molto semplice per applicare le conoscenze acquisite.

```
        List     p=16F870        ;Processore
        include  "P16F870.INC"   ;Definizione dei registri
                                  interni

        ORG      0xB0

inizio  clrf     PORTB           ;Azzera i valori casuali
                                  dato che la utilizzeremo
                                  come uscita
        bsf      STATUS,RP0      ;Seleziona banco 1
        clrf     TRISC           ;Porta C configurata
                                  come uscita
        movlw    0xFF
        movwf    TRISB           ;Porta B configurata
                                  come ingresso
        bcf      STATUS,RP0      ;Seleziona banco 0

Loop    movf     PORTB,W         ;Legge gli ingressi
        movwf    PORTC           ;Li porta sull'uscita
        goto     Loop            ;Ciclo senza fine

        end                      ;Fine del programma
```

Configurazione delle porte per risolvere l'esempio.

tivo di avere un ingresso per realizzare un'acquisizione, poter offrire il risultato di una comparazione o fornire un'uscita in modo PWM. Infine, tramite la porta C, è anche possibile la comunicazione in modo asincrono e sincrono. Il terminale RC6/TX/CK ha multiplexate la funzione di I/O digitale con quella di trasmissione seriale in modo sincrono per il modulo USART e quella di ricezione degli impulsi di clock che segneranno il sincronismo nel trasferimento seriale in modo sincrono. Il terminale RC7/RX/DT può lavorare come I/O digitale, come ricevitore di dati in modo asincrono per la USART e come I/O di trasferimento seriale in modo sincrono.

Architettura interna della porta C

I terminali della porta C hanno buffer di ingresso Trigger di Schmitt. Nella figura della pagina precedente è possibile osservare lo schema interno di ogni linea della porta C. Mediante il bit di selezione della funzione del terminale diamo l'ordine al multiplexer per fare in modo che il terminale funzioni come I/O digitale, ovvero a disposizione della periferica. Il segnale periferico OE si attiva solamente quando vogliamo lavorare con il dispositivo interno; si tratta di un segnale di abilitazione che attiva l'uscita della periferica interna. Quando, con la porta C, si lavora con un dispositivo interno, que-

Indirizzo	Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valore in POR o in BOR	Valore negli altri reset
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	Configurazione dei pin della porta C								11111111	1111 1111
x: sconosciuto, u: non cambia											

Registri associati alla porta.

st'ultimo può richiedere che il terminale a cui fa riferimento sia configurato come ingresso oppure, in altri casi, come uscita e per questo il progettista dovrà prestare particolare attenzione nel configurare il registro TRISC in modo adeguato. In molti casi dovremo modificare il valore di questo registro diverse volte nel corso di un programma.

Esempio pratico

Supponiamo di dover eseguire un progetto in cui si voglia sviluppare un sistema di allarme grazie al quale il vigilante di un museo possa controllare, da un posto di guardia, le sei sale di cui è composto il museo.

Se montiamo un sensore o un rilevatore di presenza in ogni sala, questi darà un segnale digitale quando rileverà un intruso. Avremo bisogno di sei ingressi digitali da dedicare ai sensori e, ad esempio, sei uscite digitali collegate a dei diodi LED che indicheranno in quale sala è stato generato il segnale di allarme.

Se osserviamo come sono state configurate le porte, vedremo che con la porta B avremo otto ingressi digitali e, grazie alla porta C, potremo fornire otto uscite digitali. Quindi l'unica cosa che dovrà fare il programma sarà quella di leggere continuamente gli ingressi e passare il valore letto alle uscite. Il sistema può essere ampliato, dato che ci avanzano due ingressi e due uscite. Possiamo controllare così, due variabili in più e lavorare su due elementi diversi. È possibile anche complicare un po' il programma in modo che, mediante temporizzatori, si attivi l'allarme se il segnale si mantiene per un determinato tempo, creare un allarme intermittente, attivare un cicalino, ecc.

Linea di codice	movf	PORTC,W	
Token 1	mov	Istruzione	Istruzione del processore
Token 2	PORTC	Operando	Registro
Token 3	W	Operando	Registro

Esempio tokens in una linea di codice.

Errori nella programmazione

Quando si realizza un programma, è facile commettere errori. Dobbiamo prestare attenzione quando programmiamo, perché molte volte mettere a punto un programma risulta un lavoro molto complicato. L'utilizzo di una metodologia di programmazione è un buon aiuto per evitare gli errori e per rilevarli o eliminarli, nel caso ne avessimo fatto qualcuno.

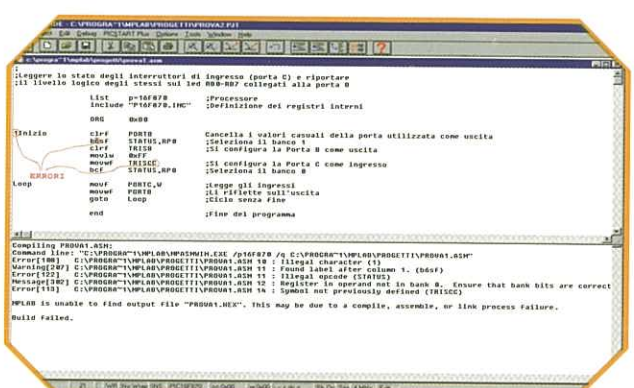
Molti compilatori ci indicano se abbiamo commesso errori, dove e di che tipo, però un compilatore non sa qual è lo scopo del nostro programma, non sa se ciò che sta traducendo in linguaggio macchina farà realmente quello che il programmatore desidera fare.

Errori lessicali, sintattici e semantici

Gli errori che possiamo commettere programmando si possono classificare in tre tipi: lessicali, sintattici e semantici.

Livello lessico

Quando si parla del livello lessico si fa riferimento alla minore entità possibile da analizzare ovvero, nel nostro caso, i simboli e i carat-

Programma con errori lessicali.

Programma con errori di sintassi.

Diversi livelli di analisi.

Programma con errori semantici.

teri. Quando utilizziamo un simbolo non permesso dal linguaggio di programmazione, stiamo commettendo un errore lessicale. Nel programma della figura sono stati inseriti caratteri che non sono accettati dal compilatore. Come potrete verificare, mediante il compilatore è possibile identificare la linea dove è stato commesso l'errore e in cosa consiste quest'ultimo. Quando il compilatore rileva un errore non esegue la traduzione in codice macchina, e in questo esempio ne sono stati trovati tre. Il compilatore avvisa anche di possibili situazioni che potrebbero provocare un funzionamento anomalo del programma, gli avvisi o warning non impediscono la compilazione.

Livello sintattico

Una combinazione di simboli raggruppati creerà ciò che si chiama un token, ovvero ognuna delle parti di una linea di codice in assembler. Nella figura della pagina precedente possiamo vedere un esempio di ciò che si intende per token. Gli errori lessicali si verificano quando l'associazione di simboli o caratteri non creano token validi.

Osservate ciò che dice il compilatore quando cerchiamo di compilare il programma della figura. Il primo errore lo dà il token "1Inizio" che è un'etichetta e come tale non può iniziare con un numero. Il secondo errore si trova sulla linea di codice successiva con il token "bt6s". il compilatore ci informa che si tratta di un'istruzione non valida, codice OP illegale. Il terzo errore ce lo dà quando stiamo indirizzando un'operazione a un registro inesistente (TRISCC).

Livello semantico

Il raggruppamento di diversi token crea una struttura, una linea di codice, che ha un significato. Nel livello semantico si vuole analizzare il significato delle strutture. Il primo errore che contiene il programma è sulla linea 11, per errore di operando: stiamo utilizzando un'istruzione che gestisce un bit, forniamo il registro su cui si trova questo bit, ma non il bit su cui deve agire.

Il compilatore ci avvisa mediante una warning dell'errore successivo di programmazione, che consiste nel fatto che il valore indicato nell'operando è fuori range (si vuole inserire un valore di 12 bit –FFFh- su un registro da 8 bit –W-). Le warning non impediscono al compilatore di tradurre il codice in linguaggio macchina. L'errore successivo ci verrà indicato con un messaggio. Il compilatore nei messaggi ci informa delle cose che non capisce ma che sono corrette, e che possono essere volontà del programmatore; praticamente consiste in un semplice: Sei sicuro di fare questo? Infine, abbiamo commesso un altro errore, perché l'istruzione situata sulla linea 19 (goto) accetta solamente un operando, un solo token dopo di essa. Corretti gli errori e compilato il programma potrebbe succedere che quest'ultimo non lavori come realmente desideriamo. L'errore in molti di questi casi si può trovare in un'errata impostazione iniziale del progetto.

Errore	Errore inaccettabile per il compilatore	NON compila
Warning	Avviso di possibile errore o incongruenza nel codice	Compila
Message	Messaggio al programmatore che avvisa di una possibile situazione di errore	Compila

Messaggi del compilatore di MPLAB.